Computing and Informatics, Vol. 41, 2022, 1054–1077, doi: 10.31577/cai_2022_4_1054

REPAIRING PROCESS MODELS WITH NON-FREE-CHOICE CONSTRUCTS BASED ON TOKEN REPLAY

Erjing Bai

Qingdao Huanghai University
Qingdao 266427, China

Man Qi

School of Engineering, Technology and Design
Canterbury Christ Church University
CT1 1QU, UK

Wenjing Luan*, Peng Li*, Yuyue Du

College of Computer Science and Engineering
Shandong University of Science and Technology
Qingdao 266590, China
e-mail: 1832678460@qq.com, yydu001@163.com

Abstract. A method of repairing process models with non-free-choice constructs is proposed based on logical Petri nets, aiming at the problem of low precision in the existing repair methods. An extended successor matrix of transitions is determined according to the distance between any two transitions. There are two types of choice-construct transitions. One is a non-free-choice construct transition, and the other is a general choice construct transition. The type of choice-construct transitions can be determined based on the extended successor matrix and the relationship between the front and back sets of transitions. The location of the deviations is calculated by an improved replaying method. Finally, a model can be

* Corresponding author

repaired according to remaining-token places and missing-token places. Based on the experiments on real event logs, the method proposed in this paper has a better performance in fitness, precision, and simplicity compared with its peers.

Keywords: Model repair, non-free-choice constructs, logical Petri net, token replay, process model, event logs

1 INTRODUCTION

Process mining is a new emerging discipline and can improve and perfect the processes in various application fields [1]. One of the foundations of process mining is data mining. However, it is different from data mining. They are data-driven mining technologies essentially, and can analyze data and find out some valuable information. There are some differences between them, such as the field of use, algorithm technology, and storage mode. Process mining based on event logs mines real process models from different perspectives. It can find the similarities and differences with standard process models and provide a basis for improving the model. Data mining does not focus on the process and cannot realize the end-to-end process discovery. It focuses on the valuable information of the data. Process mining can use Play-In, Play-Out, and Replay to create valid associations between the process model and the "reality" captured from the event logs. It uses Petri nets as input information for Play-Out, and it finally generates its described behaviors by making tokens repeatedly execute on the net. Play-In is the opposite of Play-Out. It uses behaviors as input information and a process model as output information for Play-In. All process discovery methods belong to the case of Play-In technology [1, 2]. Replay takes event logs and process models as inputs according to different purposes. Event logs are re-executed on the process model in the operation process. The deviations between event logs and process models can be detected and quantified by replaying the event logs.

Many process mining algorithms have been proposed, since the concept of Petri nets was founded by German scientist Dr. Carl Adam Petri in 1962. A process model can be mined based on α algorithm [3] according to the order of event logs. However, the deficiency of α algorithm is that the logs are required to be complete. α # is an improved algorithm of α algorithm for solving the problem of invisible transitions [4]. An extended method of α algorithm is proposed in [5] for dealing with non-free-choice constructs. The integer linear programming (ILP) algorithm is proposed in [6]. It can solve the short loop mining problem to a certain extent, while it is slow for processing event logs, so its efficiency is low. A new process model can be mined from event logs by mining algorithms. However, some of the existing models cannot be repaired by mining algorithms. Many existing process models are inconsistent with event logs, they need to be repaired. Hence, it is not enough to rely only on mining algorithms. A similar optimal alignment computing

method is proposed based on Petri nets with basic structures in [7], which does not focus on process repairing. Fahland et al. propose a repairing method with high fitness but low precision [8, 9]. Goldratt's method adds self-loops when repairing the model, which increases the complexity of the model [10]. A testability modeling method based on colored generalized stochastic Petri nets is proposed in [11]. It can solve the problem when failure modes are ignored in the existing testability mode. A repairing method for concurrent event process models based on Petri nets is proposed in [12]. The repaired model can completely replay the given event log and can avoid the occurrence of redundant activities caused by the loop. Sheng proposes a modular process model repairing method based on the behavioral profile in [13], and a process model is divided into five small modules by a divide and conquer method. A process variant merging method based on Petri nets is proposed in [14] to recognize the common characteristics of existing business flows and to eliminate the process redundancy.

Logical Petri nets [15, 16, 17] are high-level abstractions or extensions of Petri nets and high-level Petri nets. The uncertainty of logical transitions is restricted by logical expressions. Therefore, logical Petri nets can effectively avoid invisible transitions and self-loops. Many methods based on logical Petri nets for repairing and mining models have been proposed. Zheng's method [15] is a model repair method for non-free-choice structures based on logical Petri nets, and it is improved by the proposed method in this paper. An extended colored logical Petri net is proposed in [16]. Li et al. [17] analyze the application of logical Petri nets to E-commerce systems. A soundness checking method based on logical Petri net modeling for a file merging process is proposed in [18]. Guan et al. [19] propose modeling and analysis of parking reservation system based on logical time delay Petri nets to solve the problem that the logical Petri net cannot fully describe the time of transitions.

In this paper, a new model repairing method by token replaying based on logical Petri nets is proposed for non-free-choice constructs. The proposed method has high fitness and precision compared with Fahland's method [8] and Goldratt's method. Fahland's method [8] and Goldratt's method focus on the fitness while pay insufficient consideration for simplicity and precision. Compared with Zheng's method [15], the proposed method has low time complexity, and they have similar simplicity, precision and fitness. The contribution of this paper is as follows.

1. Event logs are forced to fire by the improved replaying algorithm in the workflow nets, and the deviations between an original model and event logs are found. The repairing positions can be determined by the deviations.
2. Several basic concepts are defined such as extended successor, extended successor matrix and set of choice relationship to repair an original model. The type of choice relationship transitions is distinguished by the number of pre-set transitions and post-set transitions. Finally, an original model can be repaired according to different choice relationship types.
3. A simulation experiment is conducted in this paper. The process model and the event logs used in the experiment are produced from a hospital in Qingdao.

Experimental results show that the proposed repairing method has high fitness and precision compared with some other methods.

The rest of this paper is organized as follows. Section 2 presents some basic concepts on multi-set, trace, event log, pre-set, post-set, Petri nets, logical Petri nets, and workflow nets. Section 3 proposes an approach to repair models with non-free-choice constructs via logical Petri nets. Simulation experiments are carried out and the analysis of the experimental results is given in Section 4. Section 5 summarizes this paper and discusses the future work.

2 PRELIMINARIES

Some basic concepts are reviewed and briefly introduced in this section including multi-sets [20], event logs [21, 22], pre-set, post-set [15, 16], Petri nets [23, 24, 25], logical Petri nets [16, 17], and workflow nets [26].

Definition 1 (Multi-Sets [20]). Ψ is a set. A multi-set D over Ψ is denoted by $D : \Psi \to N^+$, where N^+ represents a set of positive integers. All multi-sets over Ψ are denoted by $\beta(\Psi)$.

For example, for a multi-set D over Ψ, $D = [b^2, c^3, d]$ where $b, c, d \in \Psi$, $D(b) = 2$, $D(c) = 3$, and $D(d) = 1$. $D_1, D_2 \in \beta(\Psi)$ are two multi-sets. $D_3 \in \beta(\Psi)$ is the union of D_1 and D_2, $D_3 = D_1 \biguplus D_2$, and for $\forall b \in \Psi : D_3(b) = D_1(b) + D_2(b)$. $D_4 = D_1 \setminus D_2$ is the difference of D_1 and D_2, where $\forall a \in \Psi : D_4(a) = \max\{0, D_1(a) - D_2(a)\}$.

Definition 2 (Sequence). Let A be a set. $w = \langle w[1], w[2], \ldots, w[n] \rangle$ is a sequence over A, where $w[i] \in A$ $(1 \leq i \leq n)$ denotes the i^{th} element of w.

Definition 3 (Trace [27, 28, 29]). A is a set of activities. A trace $\sigma \in A^*$ is a sequence of activities, and $|\sigma|$ represents the length of σ, where $1 \leq i < j \leq |\sigma| : \sigma[i] \neq \sigma[j]$.

Definition 4 (Event log [21, 22]). A is a set of activities. $\sigma \in A^*$ is a trace. An event log denoted by $L \in \beta(A^*)$ is a finite multi-set over σ. $\&(\sigma)$ represents a collection of all activities of σ.

For example, given an activity set $A' = \{t_1, t_2, t_3, t_4\}$, $\sigma' = \{t_2, t_3, t_4\}$ is a trace, and $L' = \{\langle t_2, t_3, t_4 \rangle^6, \langle t_1, t_2, t_3, t_4 \rangle^7\}$ is an event log with 13 traces. There are $3 \times 6 + 4 \times 7 = 46$ events in total.

Definition 5 (Net [21, 22]). $N = (P, T; F)$ is a net, where

1. P is a finite set of places;
2. T is a finite set of transitions, and $T \cap P = \phi$, $T \cup P \neq \phi$; and
3. $F \subseteq (P \times T) \cup (T \times P)$ is a directed arcs set.

Definition 6 (Pre-Set and Post-Set [16]). Let $N = (P, T; F)$ be a net. For $x \in P \cup T$, ${}^\bullet x = \{y | y \in P \cup T \wedge (y, x) \in F\}$ denotes the pre-set of x, $x^\bullet = \{y | y \in P \cup T \wedge (x, y) \in F\}$ denotes the post-set of x, and ${}^\bullet x \cup x^\bullet$ denotes the extension of x.

For example, a net $N_1 = (P, T; F)$ is shown in Figure 1, where $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$, $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$, and $F = \{(p_1, t_1), (p_1, t_2), (p_2, t_3), (p_2, t_4), (p_3, t_5), (p_4, t_6), (p_5, t_7), (p_6, t_7), (t_1, p_2), (t_2, p_2), (t_3, p_3), (t_4, p_4), (t_5, p_5), (t_6, p_6), (t_7, p_1)\}$. For the net N_1, the pre-set and post-set of ${}^\bullet p_1 = \{t_7\}$, $p_1^\bullet = \{t_1, t_2\}$. The pre-set and post-set of ${}^\bullet t_7 = \{p_5, p_6\}$, $t_7^\bullet = \{p_1\}$.

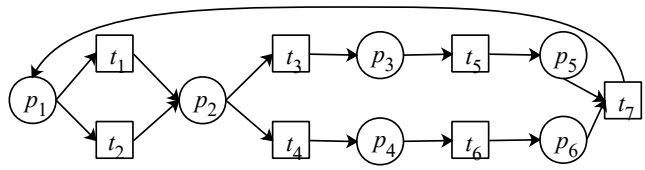


Figure 1. A net model N_1

Petri nets are a type of bipartite-directed graphs. The graph has three types of objects named places, transitions, and directed arcs connecting places to transitions and transitions to places [23, 24, 25], respectively.

Definition 7 (Petri net [23, 24, 25]). $PN = (P, T; F, M)$ is a Petri net. P and T represent a finite set of places and transitions, respectively. $F \subseteq (P \times T) \cup (T \times P)$ is a finite arc set. $M : P \to \{0, 1, 2, \dots\}$ is a marking of PN, and the rules of transition firing are as follows:

1. $M[t\rangle$ represents that transition $t \in T$ is enabled under M, where $\forall p \in^\bullet t : M(p) \geq 1$;
2. If $M[t\rangle$, t can be fired, and when t fires, a new marking M' is generated, denoted as $M[t\rangle M'$, where for $\forall p \in P$, we have

$$M'(p) = \begin{cases} M(p) - 1, & p \in^\bullet t - t^\bullet; \\ M(p) + 1, & p \in t^\bullet -^\bullet t; \\ M(p), & \text{otherwise.} \end{cases}$$

A workflow net is a special Petri net that has been widely used in modeling business process management systems [30, 31]. The soundness of workflow nets is an important criterion. It has been proven that the soundness problem is decidable [32, 33, 34]. For the correctness of workflow systems, some methods are developed to detect and repair errors, such as literature [35].

Definition 8 (Workflow net [26]). $WFN = (P, T; F, M, i, o)$ denotes a workflow net, where

1. P, T, F and M are constituted a Petri net, and the meaning of P, T, F and M is the same as Definition 7;
2. $i \in P$ is an input place with ${}^\bullet i = \phi$, and M_i denotes the initial marking where $M_i = 1$ and others are 0;
3. $o \in P$ is an output place with $o^\bullet = \phi$, and M_o is the final marking where $M_o = 1$ and others are 0; and
4. $\forall x \in P \cup T$ is on the path from i to o.

Definition 9 (Logical Petri net [16, 17]). $LPN = (P, T; F, I, O, M)$ is a logical Petri net, where

1. P is a finite set of places;
2. T is the union of T_D, T_I and T_O, which denotes a finite transition set, and $T \cap P = \phi$. T_D is the same as Definition 7, which represents a set of conventional transitions in Petri nets. T_I and T_O represent input and output transition sets, respectively. For $\forall t \in T_I$, $f_I(t)$ is a logical input expression that restricts ${}^\bullet t$; and for $\forall t \in T_O$, $f_O(t)$ is a logical output expression that restricts $t^\bullet$;
3. $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs;
4. I is a mapping of $f_I(t)$, and for $\forall t \in T_I$, $I(t) = f_I(t)$;
5. O is a mapping of $f_O(t)$, and for $\forall t \in T_O$, $O(t) = f_O(t)$;
6. $M : P \to \{0, 1, 2, \dots\}$ is a marking function of LPN, and the firing rules are as follows:
   (a) For $\forall t \in T_D$, the transition firing rules remain the same with a Petri net;
   (b) For $\forall t \in T_I$, if $f_I(t)|_M =_\bullet T_\bullet$, then a logical input transition of t can fire, denoted as $M[t > M'$, and for $\forall p \in^\bullet t$, $M'(p) = 0$; and for $\forall p \notin^\bullet t \cup t^\bullet$, $M'(p) = M(p)$; and for $\forall p \in t^\bullet$, $M'(p) = 1$; and
   (c) For $\forall t \in T_O$, if $\forall p \in^\bullet t$, $M(p) = 1$, then a logical output transition of t can fire, and for $\forall p \in^\bullet t$, $M'(p) = 0$. For $\forall p \in t^\bullet : f_O(t)|_M =_\bullet T_\bullet$, and for $\forall p \notin^\bullet t \cup t^\bullet : M'(p) = M(p)$.
7. There are three symbols $\otimes$, $\vee$ and $\wedge$ for $I(t)$ and $O(t)$. $p_1 \otimes p_2 \cdots \otimes p_n$ represents only one of p_1, p_2, …, p_n has tokens; $p_1 \wedge p_2 \cdots \wedge p_n$ represents each of p_1, p_2, …, p_n has tokens; $p_1 \vee p_2 \cdots \vee p_n$ represents at least one of p_1, p_2, …, p_n has tokens; where $n \geq 2$.

For example, a logical Petri net LPN_1 is shown in Figure 2. There are three transitions in LPN_1: t_1 is an input transition; t_3 is an output transition and t_2 is a traditional transition. $I(t_1) = p_1 \vee p_2$ is the logical input function. By firing t_1, one of p_1 and p_2 gets a token at least. $O(t_3) = p_5 \otimes p_6$ is the logical output function. There are two situations when t_3 is fired: p_5 or p_6 gets a token.

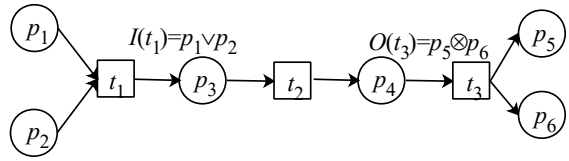


Figure 2. A logical Petri net model LPN_1

3 MODEL REPAIRING OF NON-FREE-CHOICE CONSTRUCTS

Conformance checking can detect the inconsistencies between a model and event logs by comparing and analyzing the original model with the actual event logs. There are many methods for conformance checking, such as token replay, alignment. The method based on token replay will replay all traces of the event logs on the process model. When deviations and other problems occur, the token replay will be stopped and the fitness of the remaining traces could not be calculated. The original model cannot be completely replayed by the event logs, so it needs to be repaired. A repairing method with non-free-choice constructs based on token replay is proposed next. The deviation positions between a model and event logs can be calculated by replaying an event log L in a model. The original model can be repaired via logical Petri nets according to the deviation positions. Tokens can be dynamically evolved from the initial place to the final place if there is no deviation between a model and event logs in the non-free-choice constructs. Otherwise, some transitions cannot be fired due to lacking tokens. So tokens cannot be finally changed to the final place. Whether there are deviations or not between a model with the non-free-choice constructs and event logs, an improved replaying algorithm [20] is proposed to ensure that tokens can be evolved from the initial place to the final place.

Example 1. A workflow net WFN_1 is shown in Figure 3 where $L_1 = \{\langle \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5 \rangle\} = \{\langle t_1, t_4, t_5, t_7, t_8, t_{10} \rangle, \langle t_1, t_4, t_5, t_7, t_8, t_{10}, t_9 \rangle, \langle t_2, t_4, t_5, t_7, t_8, t_{10}, t_9 \rangle, \langle t_1, t_3, t_4, t_6, t_7, t_9 \rangle, \langle t_2, t_4, t_5, t_7, t_8, t_{10} \rangle\}$. σ_1 is replayed in WFN_1, and Table 1 shows the changes of tokens based on the improved replaying algorithm [20].

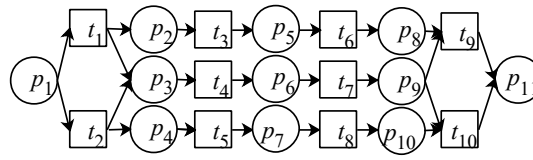


Figure 3. Workflow net model WFN_1

If $\forall i \in \{1, \ldots, 9\} : M(p_i) = 0$ at the end of replaying, it means that the trace

is completely fitted with the model; otherwise there are deviations between the trace and the model. After executing the improved replaying algorithm [20] with σ_1, $M(p_4) = -1$ and $M(p_2) = 1$ represent the missing-token and the remaining-token, respectively. For repairing the model, it is necessary to replay other traces to determine the repair locations. Thus, σ_2 is replayed in WFN_1, and $M(p_4) = -1$, $M(p_2) = 1$, $M(p_8) = p_9 = -1$, $M(p_{11}) = 1$ after replaying. It means that p_8 and p_9 are the deviations besides p_2 and p_4. For repairing non-free-choice constructs via logical Petri nets, it needs to determine the transitions set with a selection relationship. Then the following concepts are given.

Transitions	$M(p_1)$	$M(p_2)$	$M(p_3)$	$M(p_4)$	$M(p_5)$	$M(p_6)$	$M(p_7)$	$M(p_8)$	$M(p_9)$	$M(p_{10})$	$M(p_{11})$
Start	1	0	0	0	0	0	0	0	0	0	0
t_1	0	1	1	0	0	0	0	0	0	0	0
t_4	0	1	0	0	0	1	0	0	0	0	0
t_5	0	1	0	−1	0	1	1	0	0	0	0
t_7	0	1	0	−1	0	0	1	0	1	0	0
t_8	0	1	0	−1	0	0	0	0	1	1	0
t_{10}	0	1	0	−1	0	0	0	0	0	0	1
end	0	1	0	−1	0	0	0	0	0	0	0

Table 1. The change of tokens in WFN_1

Definition 10 (Extended successor)**.** A is a set of activities. $\sigma \in A^*$ is a trace. In trace σ, $\sigma[i]$ occurs before $\sigma[i+n]$, $1 \leq i \leq |\sigma - 1|$, $n \geq 1$, then $\sigma[i+n]$ is a successor of $\sigma[i]$. The extended successor is denoted by $\sigma[i] >>_n \sigma[i+n]$.

For example, $\sigma = \langle a, b, c, d \rangle$, the extended successor can be expressed as: $a >>_1 b$, $a >>_2 c$, $a >>_3 d$, $b >>_1 c$, $b >>_2 d$, and $c >>_1 d$.

Definition 11 (Extended successor distance)**.** A is a set of activities. $\sigma \in A^*$ is a trace. In trace σ, $\sigma[i] >>_n \sigma[i+n]$ where $1 \leq i \leq |\sigma - 1|$, $n \geq 1$. The extended successor distance can be expressed as: $\mathrm{dis}(\sigma[i], \sigma[i+n]) = n$ and it represents the shortest distance from $\sigma[i]$ to $\sigma[i+n]$.

For example, $L_2 = \{\langle \sigma_1, \sigma_2 \rangle\} = \{\langle a, b, c, d \rangle, \langle a, d, e \rangle\}$. The extended successor between a and d is $a >>_3 d$ and $a >>_1 d$ in trace σ_1 and σ_2, respectively. So the extended successor distance from a to d is $\mathrm{dis}(a, d) = 1$.

Definition 12 (Extended successor matrix)**.** Let $PN = (P, T; F, M)$ be a Petri net. $\forall t_i$, $t_j \in T$, $A_L(t_i, t_j)$ is an extended successor matrix of $|T|$, and $\mathrm{dis}(t_i, t_j)$ is the shortest distance from t_i to t_j.

$$A_L(t_i, t_j) = \begin{cases} n, & (t_i, t_j) \in t_i >>_n t_j; \\ 0, & \text{otherwise.} \end{cases}$$

According to the above definitions, the algorithm for calculating the extended successor matrix is given as follows.

Algorithm 1 Calculating the extended successor matrix

Input: A Workflow net $WFN = (P, T; F, i, o)$ and an event log $L \in \beta(\sigma^*)$;
Output: A_L.

1: $\sigma_L \leftarrow |\sigma_1|$;
2: **for** $(i = 0;\ i < |T|;\ i{+}{+})$ **do**
3: **for** $(j = 0;\ j < |T|;\ j{+}{+})$ **do**
4: $A_L[i][j] \leftarrow 0$;
5: **end for**
6: **end for**
7: **for** $(i = 1,\ \sigma_i \in L;\ i <= |L|;\ i{+}{+})$ **do**
8: **if** $|\sigma_i| > \sigma_L$ **then**
9: $\sigma_L = |\sigma_i|$
10: **end if**
11: **end for**
12: $TL = \sigma_L - 1$;
13: **for** $(n = 1;\ n <= TL;\ n{+}{+})$ **do**
14: **for** $(i = 1,\ \sigma_i \in L;\ i <= |L|;\ i{+}{+})$ **do**
15: **for** $(j = 1;\ (j + n) <= |\sigma_i|;\ j{+}{+})$ **do**
16: **if** $A_L[\sigma_i[j]][\sigma_i[j{+}n]] \neq 0$ and $A_L[\sigma_i[j]][\sigma_i[j{+}n]] > n$ or $A_L[\sigma_i[j]][\sigma_i[j{+}n]] = 0$ **then**
17: $A_L[\sigma_i[j]][\sigma_i[j + n]] = n$;
18: **end if**
19: **if** $(\sigma_i[j]^\bullet, \sigma_i[j + n]) \notin F$ and $(\sigma_i[j], {}^\bullet\sigma_i[j + n]) \notin F$ **then**
20: $A_L[\sigma_i[j]][\sigma_i[j + n]] = 0$;
21: **end if**
22: **end for**
23: **end for**
24: **end for**
25: **return** A_L.

In Algorithm 1, the initialization is completed in Steps 1–6. The longest trace in $\log L$ is calculated in Steps 7–11, and saved in σ_L. TL is assigned $\sigma_L - 1$ in Step 12. A_L is calculated in Steps 13–24. In Step 25, the results are returned. Algorithm 1 is simple and its computational complexity is $O(n^3)$.

Example 2. A workflow net WFN_1 is shown in Figure 3 where $L_1 = \{\langle \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5 \rangle\} = \{\langle t_1, t_4, t_5, t_7, t_8, t_{10} \rangle, \langle t_1, t_4, t_5, t_7, t_8, t_{10}, t_9 \rangle, \langle t_2, t_4, t_5, t_7, t_8, t_{10}, t_9 \rangle, \langle t_1, t_3, t_4, t_6, t_7, t_9 \rangle, \langle t_2, t_4, t_5, t_7, t_8, t_{10} \rangle\}$. Then, the extended successor matrix of event log

L_1 is as follows.

$$A_L = \begin{pmatrix} 0 & 0 & 1 & 1 & 2 & 3 & 3 & 4 & 5 & 5 \\ 0 & 0 & 0 & 1 & 2 & 0 & 3 & 4 & 6 & 5 \\ 0 & 0 & 0 & 1 & 0 & 2 & 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 & 3 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Definition 13 (Set of choice relationship)**.** Let $PN = (P, T; F, M)$ be a Petri net, and $\forall t_i,\ t_j \in T$, $A_L(t_i, t_j)$ be an extended successor matrix of $|T|$. If $\forall t_c,\ t_d \in T$, ${}^\bullet t_c \cap {}^\bullet t_d \neq \phi$, $t_c^\bullet \cap t_d^\bullet \neq \phi$, $|{}^\bullet t_c| \geq 1$, $|{}^\bullet t_d| \geq 1$ and $A_L(t_c, t_c) = A_L(t_d, t_d) = A_L(t_c, t_d) = A_L(t_d, t_c) = 0$, then t_c and t_d are non-free-choice relationship, and it is denoted by the set of $NCRT$. If $\forall t_c,\ t_d \in T$, ${}^\bullet t_c \cap {}^\bullet t_d \neq \phi$, $t_c^\bullet \cap t_d^\bullet \neq \phi$, $|{}^\bullet t_c| = 1$, $|{}^\bullet t_d| = 1$ and $A_L(t_c, t_c) = A_L(t_d, t_d) = A_L(t_c, t_d) = A_L(t_d, t_c) = 0$, then t_c and t_d have a general choice relationship, and it is denoted by the set of $CCRT$. Algorithm 2 is to calculate the sets $NCRT$ and $CCRT$.

Algorithm 2 Calculating the set of choice relationship

Input: A workflow net $WFN = (P, T; F, M, i, o)$ and the extended successor matrix denoted by A_L;

Output: $NCRT$, $CCRT$.

1: $CCRT \leftarrow \phi$, $NCRT \leftarrow \phi$;
2: **for** $(i = 1;\ i <= |T|;\ i{+}{+})$ **do**
3: **for** $(j = 1;\ j <= |T|;\ j{+}{+})$ **do**
4: **if** ${}^\bullet t_i \cap {}^\bullet t_j \neq \phi$, $t_i^\bullet \cap t_j^\bullet \neq \phi$, $|{}^\bullet t_i| = 1$, $|{}^\bullet t_j| = 1$ and $A_L(t_i, t_i) = A_L(t_j, t_j) = A_L(t_i, t_j) = A_L(t_j, t_i) = 0$ **then**
5: $CCRT = CCRT \cup t_i \cup t_j$;
6: **end if**
7: **if** ${}^\bullet t_i \cap {}^\bullet t_j \neq \phi$, $t_i^\bullet \cap t_j^\bullet \neq \phi$, $|{}^\bullet t_i| \geq 1$, $|{}^\bullet t_j| \geq 1$ and $A_L(t_i, t_i) = A_L(t_j, t_j) = A_L(t_i, t_j) = A_L(t_j, t_i) = 0$ **then**
8: $NCRT = NCRT \cup t_i \cup t_j$;
9: **end if**
10: **end for**
11: **end for**
12: **return** $CCRT$, $NCRT$.

In Algorithm 2, the initialization is completed in Step 1. In Steps 2–11, all transitions in WFN are travelled: if ${}^\bullet t_i \cap {}^\bullet t_j \neq \phi$, $t_i^\bullet \cap t_j^\bullet \neq \phi$, $|{}^\bullet t_i| = 1$, $|{}^\bullet t_j| = 1$

and $A_L(t_i, t_i) = A_L(t_j, t_j) = A_L(t_i, t_j) = A_L(t_j, t_i) = 0$, then t_i and t_j are saved in $CCRT$; if ${}^\bullet t_i \cap {}^\bullet t_j \neq \phi$, $t_i^\bullet \cap t_j^\bullet \neq \phi$, $|{}^\bullet t_i| \geq 1$, $|{}^\bullet t_j| \geq 1$ and $A_L(t_i, t_i) = A_L(t_j, t_j) = A_L(t_i, t_j) = A_L(t_j, t_i) = 0$, then t_i and t_j are saved in $NCRT$. In Step 12, the last results are returned.

In Algorithm 3, the initialization is in Step 1. It calls Algorithm 1 to calculate the extended successor matrix in Step 2. It calls Algorithm 2 to calculate the set of choice relationship $CCRT$ or $NCRT$ in Step 3. The trace σ_i in event log L is replayed to determine the repair locations. The results are saved in P_r and P_m for the remaining-token places and missing-token places in Steps 4–14. The model is repaired in Steps 15–46. It is to identify whether the front set of the remaining-token places belongs to a general choice relationship or a non-free-choice relationship in the first step. If ${}^\bullet P_r$ belongs to the general choice relationship $CCRT$, it needs to add an arc from ${}^\bullet P_x$ to P_y and add a logical output function $O({}^\bullet P_x) = O(t_a) \otimes O(t_b)$ for repairing the model. Otherwise, it needs to add a new place p', two arcs $p' \to t_n$, $t_m \to p'$, and two logical functions $O(t_m)$, $I(t_n)$. At last, it returns the logical Petri net LPN in Step 47. The computational complexity of this algorithm is $O(n^3)$.

Example 3. Workflow net WFN_1 can be repaired based on Algorithm 3, where $L_1 = \{\langle \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5 \rangle\} = \{\langle t_1, t_4, t_5, t_7, t_8, t_{10} \rangle, \langle t_1, t_4, t_5, t_7, t_8, t_{10}, t_9 \rangle, \langle t_2, t_4, t_5, t_7, t_8, t_{10}, t_9 \rangle, \langle t_1, t_3, t_4, t_6, t_7, t_9 \rangle, \langle t_2, t_4, t_5, t_7, t_8, t_{10} \rangle\}$. The repaired model is shown in Figure 4. According to Algorithm 3, the repairing process is given as follows.

1. Calling Algorithms 1 and 2 to calculate the extended successor matrix A_L and $CCRT$, $NCRT$, respectively. The result of A_L is the same as Example 2, and $CCRT = \{t_1, t_2\}$, $NCRT = \{t_9, t_{10}\}$.
2. σ_1 is replayed in WFN_1 based on the improved replaying algorithm [20]. It makes $M(p_4)$ equal to -1 and $M(p_2)$ equal to 1. Then they are saved in the set of P_m and P_r, respectively.
3. The model is repaired according to the above results. The arc from ${}^\bullet P_r$ to P_m is added because of ${}^\bullet P_r$. And the logical output function $O(t_1) = (p_3 \wedge p_4) \otimes (p_2 \wedge p_3)$ is added too.
4. p_2 is saved in the set of P_f.
5. σ_2 is also replayed in WFN_1 based on the improved replaying algorithm [20], and the results are $M(p_2) = M(p_{11}) = 1$, $M(p_4) = M(p_8) = M(p_9) = -1$. They are also saved in the set of P_r and P_m, respectively.
6. The deviation location of p_2 is skipped because of P_f. A new place p' is added because of p_{11}. Two arcs $p' \to t_9$, $t_{10} \to p'$ and two logical functions $O(t_{10}) = p_{11} \otimes p'$, $I(t_9) = (p_8 \wedge p_9) \otimes p'$ are added.
7. σ_3, σ_4, and σ_5 are also replayed, but the results are already saved in P_f. So the model is not repaired with the deviation which produced by σ_3, σ_4, and σ_5.
8. Finally, we can obtain a repaired model shown in Figure 4. We can see that the repaired model of WFN_1 by our method does not change its original model

Algorithm 3 Model repairing method for non-free-choice constructs

Input: A workflow net $WFN = (P, T; F, M, i, o)$ and an event log $L \in \beta(\sigma^*)$;

Output: A repaired logical Petri net, denoted by $LPN = (P, T; F, I, O, M)$.

1: $LPN \leftarrow WFN$, $O(T_a) \leftarrow \phi$, $O(T_b) \leftarrow \phi$, $P_f \leftarrow \phi$;
2: Calling Algorithm 1 to calculate the extended successor matrix A_L;
3: Calling Algorithm 2 to calculate $CCRT$ and $NCRT$;
4: **for** $(i = 1; \sigma_i \in L; i{+}{+})$ **do**
5: $P_r \leftarrow \phi$, $P_m \leftarrow \phi$;
6: Calling the improved replaying algorithm to replay the trace σ_i in $\log L$
7: **for** $(j = 1; j \leq |\sigma_i|; j{+}{+})$ **do**
8: **if** $M(p_j) > 0$ **then**
9: $P_r = P_r \cup p_j$;
10: **end if**
11: **if** $M(p_j) < 0$ **then**
12: $P_m = P_m \cup p_j$;
13: **end if**
14: **end for**
15: **for** each $P_x \in P_r$ and $P_y \in P_m$ **do**
16: **if** ${}^\bullet P_x \in CCRT$ and ${}^\bullet P_y \in CCRT$ and $P_x \notin P_f$ **then**
17: $F = F \cup^\bullet P_x \rightarrow P_y$;
18: **for** $(k = 1; k \leq |T|; k{+}{+})$ **do**
19: **if** $A_L[{}^\bullet P_x, T_k] == 1$ **then**
20: $O(T_a) \leftarrow O(T_a) \wedge^\bullet T_k$;
21: **end if**
22: **end for**
23: $A_L[{}^\bullet P_x, P_y^\bullet] = 1$;
24: **for** $(m = 1; m \leq |\sigma_i|; m{+}{+})$ **do**
25: **if** $A_L[{}^\bullet P_x, T_m] == 1$ and $T_m \in \sigma_i$ **then**
26: $O(T_b) \leftarrow O(T_b) \wedge^\bullet T_m$;
27: **end if**
28: **end for**
29: $O({}^\bullet P_x) \leftarrow O(T_a) \otimes O(T_b)$;
30: **end if**
31: **if** ${}^\bullet P_x \in NCRT$ and $P_x \notin P_f$ **then**
32: **if** $\forall P_y^\bullet ==^\bullet P_x$ **then**
33: $T_n \leftarrow^\bullet P_x$;
34: **end if**
35: $T_m \leftarrow NCRT - T_n$;
36: add a new place p'
37: $P = P \cup p'$, $F = F \cup p' \rightarrow T_n$;
38: $F = F \cup T_m \rightarrow p'$, $O(T_m) = P_x \otimes p'$;
39: **for** each ${}^\bullet T_n == P_y$ **do**
40: $I(T_n) = I(T_n) \wedge P_y$;
41: **end for**
42: $I(T_n) = I(T_n) \otimes p'$;
43: **end if**
44: **end for**
45: $P_f = P_f \cup P_x$
46: **end for**
47: **return** LPN.

structures. One place and two arcs are added to the repaired model. And it does not add self-loops and invisible transitions.

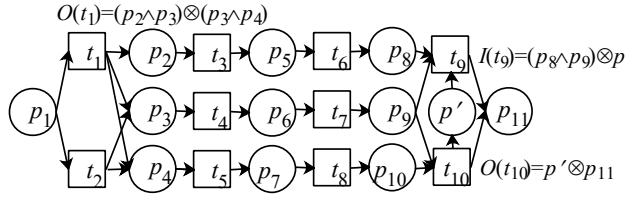


Figure 4. Repaired model of WFN_1 by our method

The models repaired by Fahland's method and Goldratt's method are shown in Figure 5 and Figure 6, respectively. In Figure 5, ten directed arcs and three invisible transitions are added to the model. Two self-loops, an invisible transition, and seven directed arcs are added to the model in Figure 6. Therefore, the repairing method proposed in this paper is more concise.

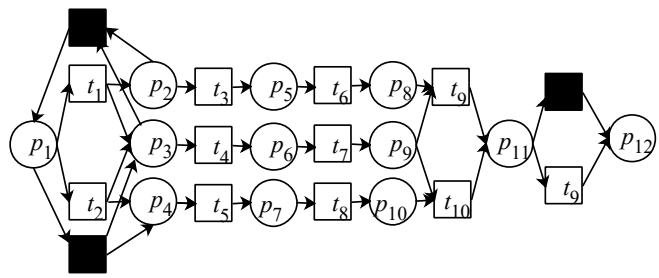


Figure 5. Repaired model of WFN_1 by Fahland's approach

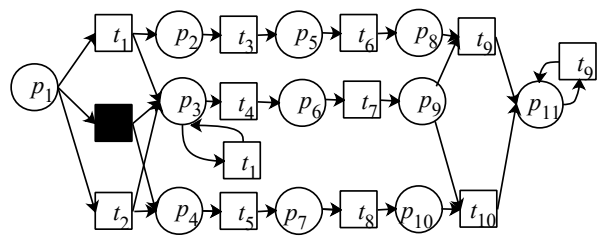


Figure 6. Repaired model of WFN_1 by Goldratt's method

4 SIMULATION EXPERIMENTS

The proposed method in this paper and other three repairing methods are simulated in this section. The experimental results are compared and analyzed. The event logs and process model used in the experiment are from the department of orthopedics of a hospital in Qingdao, and they can be accessible at: `https://pan.baidu.com/s/17lFqUck3elCiNm2X0iRjhQ?pwd=xsp5`. The repaired method of Fahland [8] is implemented in ProM 6.10 obtained at `http://www.promtools.org/`. Goldratt's method [10] is achieved in the DOS window and processed in ProM 6.10. The model repairing and analysis of our repair approach and Zheng's method [15] use manual simulation in this paper, as there are no corresponding experimental tools for mining and repairing logical Petri nets.

The process model in Figure 7 shows a series of activities of patients in the orthopedic department [15]. The activities range from making an appointment to leaving the hospital. Table 2 shows the mapping relationship between transitions and activities in Figure 7. The specific process is shown as follows. If the patient is not an emergency case, she or he needs to make an appointment at the triage station first, then queue up to register. At this point, she or he can choose a general clinic or a specialist clinic. After that, the doctor will make inquiries according to the registration order, and make corresponding examinations for patients. The selection after diagnosis will affect the choice of the non-free-choice construct subsequently. If the patient has a severe problem, she or he needs to do more detailed examinations. She or he will be hospitalized, pay, reimburse and finally leave the hospital. Otherwise, she or he needs to do basic treatment, pay the fee, get the medicine from the pharmacy with the prescription issued by the doctor, and then leave the hospital. However, the actual business process is more complex than Figure 7 according to the actual event logs. For example, patients do not want to be hospitalized after more detailed examinations, or their condition is mild, and they do not need to be hospitalized. With the continuous reform of medical insurance, some drugs can also be reimbursed, etc. Thus, the model needs to be repaired.

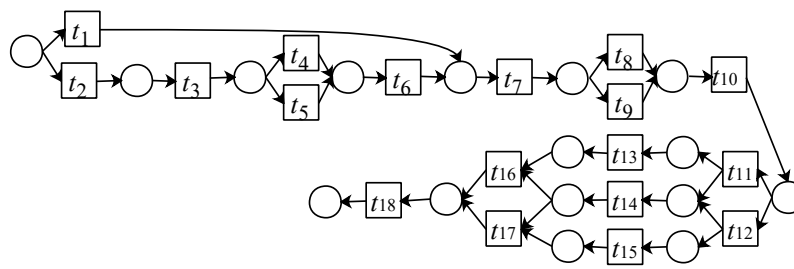


Figure 7. A Petri net of hospital orthopedic department

Transitions	Activities	Transitions	Activities
t_1	Emergency case	t_{10}	Diagnosis
t_2	Reservation at triage station	t_{11}	Further examination
t_3	Queue up	t_{12}	Basic treatment
t_4	General clinic	t_{13}	Hospitalization
t_5	Specialist clinic	t_{14}	Payment
t_6	Call number by order	t_{15}	Prescribe
t_7	Doctor consultation	t_{16}	Apply for reimbursement
t_8	Computed Tomography	t_{17}	Pharmacy taking medicine
t_9	Magnetic Resonance Imaging	t_{18}	Leaving hospital

Table 2. The mapping relationship between transitions and activities in Figure 7

The event logs shown in Table 3 are in the XES format and are preprocessed. Activities that deviated seriously from the actual business process are deleted through preprocessing for the logs. Table 3 includes the number of traces, events, activities, and the length range of traces.

Figures 8, 9 and 10 show the results of three different methods to repair the original process model according to the event logs in Table 3. It is Fahland's method [8] shown in Figure 8, provided by the toolkit ProM 6.10. We can see that the repaired model by Fahland's method adds 2 invisible transitions and 1 self-loop comparing with its original model. Goldratt's method [10] is in Figure 9, and it is achieved in the DOS window. The repaired model by Goldratt's method adds 2 self-loops and one invisible transition. The two repaired models which have some invisible transitions or self-loops have low precision.

The repairing method based on the logical Petri nets proposed in this paper is in Figure 10. We can see that the repaired model based on logical Petri nets maintains its original model structures. It adds 3 arcs, 1 place, 2 logical output functions, and one logical input function. The repaired model does not add self-loops and invisible transitions. Two logical output functions are $O(\text{Further examination}) = (p_9 \wedge p_{10}) \otimes (p_{10} \wedge p_{11})$, $O(\text{Pharmacy taking medicine}) = p_{15} \otimes p_{17}$, and the one logical input function is $I(\text{Apply for reimbursement}) = (p_{12} \wedge p_{13}) \otimes p_{17}$.

The comparison of the results by four repairing approaches is in Table 4. From Table 4, 2, and 1 invisible transitions are added by Fahland's and Goldratt's methods, respectively, while it is not added by our method and Zheng's method. From the number of arcs, 8 and 7 arcs are added by Fahland's and Goldratt's methods, respectively, while 3 arcs are increased by our method and Zheng's method. There are no repeated transitions added by our approach and Zheng's method, while Fahland's method and Goldratt's method add 1 and 2 repeated transitions, respectively. From the analysis, it can be concluded that our approach is simpler than Fahland's method and Goldratt's method, and it is the same in comparison with Zheng's method.

There are four metrics to measure the quality of a process model. They are fitness [9], precision, simplicity, and generalization.

Event Logs	Number of Traces	Number of Events	Number of Activities	Length Range of Traces
L_1	136	1 622	18	$9 \sim 13$
L_2	232	2 841	18	$9 \sim 13$
L_3	334	4 112	18	$9 \sim 13$
L_4	452	5 504	18	$9 \sim 13$
L_5	523	6 354	18	$9 \sim 13$
L_6	622	7 509	18	$9 \sim 13$
L_7	739	8 843	18	$9 \sim 13$
L_8	869	10 430	18	$9 \sim 13$
L_9	954	11 480	18	$9 \sim 13$
L_{10}	1 043	12 597	18	$9 \sim 13$
L_{11}	1 169	14 087	18	$9 \sim 13$
L_{12}	1 235	14 860	18	$9 \sim 13$
L_{13}	1 359	16 326	18	$9 \sim 13$
L_{14}	1 456	17 428	18	$9 \sim 13$

Table 3. Event logs

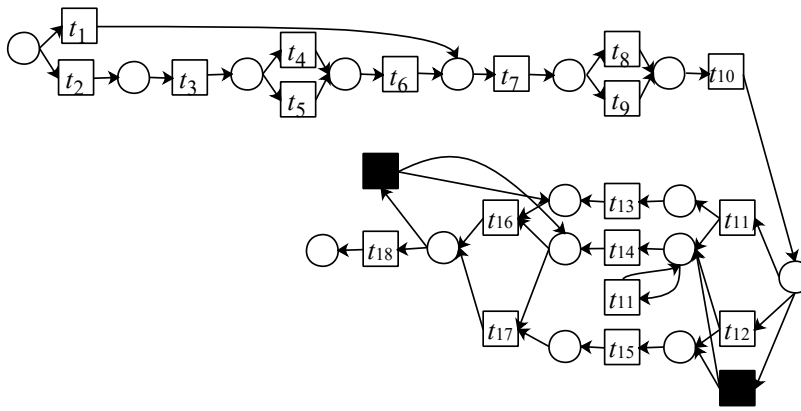


Figure 8. The repaired model by Fahland's method

Repairing Methods	Added $\lvert T + \tau \rvert$	Added $\lvert F \rvert$	Added Repeat $\lvert T \rvert$	Added $\lvert P \rvert$
Our approach	0	3	0	1
Fahland's method	2	8	1	0
Goldratt's method	1	7	2	0
Zheng's method	0	3	0	1

Table 4. Comparison of the result by four repairing approaches

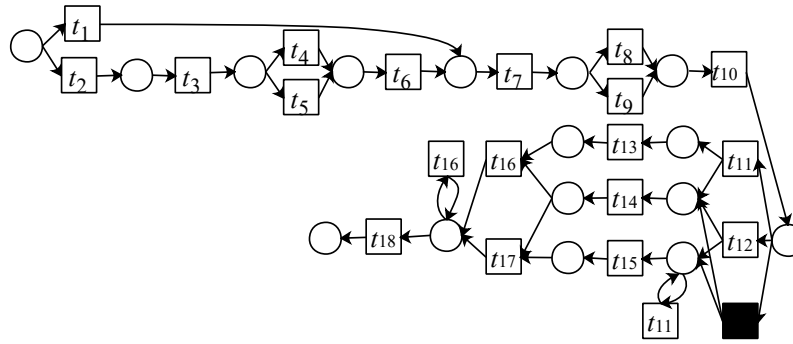


Figure 9. The repaired model by Goldratt's method

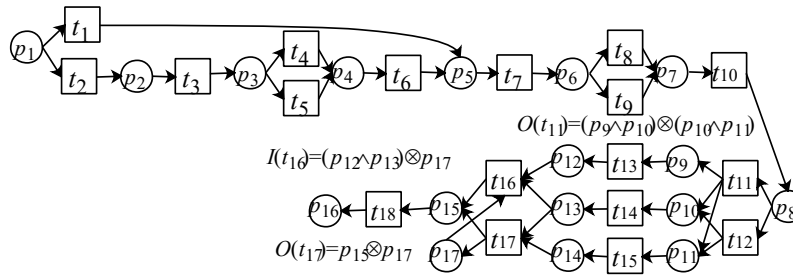


Figure 10. The repaired model by our approach

Fitness is the most important indicator, and indicates whether the event logs could be replayed by a process model. A well-fitted model allows the event logs to be replayed as much as possible. The fitness among the four different repairing methods is analyzed in Figure 11. It is calculated by the tool of ProM 6.10 for Fahland's and Goldratt's methods, and it is calculated manually according to reference [9] for our approach and Zheng's method. From Figure 11, the four different repairing methods have high fitness.

Precision is another important metric for evaluating the process model. A process model with high precision means that the model activities should be related to those in event logs. The precision among the four different repairing methods is analyzed in Figure 12. It is calculated by using the tool of "Check Precision based on Align-ETConformance" in ProM 6.10 for Fahland's and Goldratt's methods, calculated manually according to reference [9] for our approach and Zheng's method. The precision of Goldratt's and Fahland's methods is about $0.76 \sim 0.79$. The precision of our approach and Zheng's method is about 0.9. Therefore, our approach has higher precision than Fahland's and Goldratt's methods.

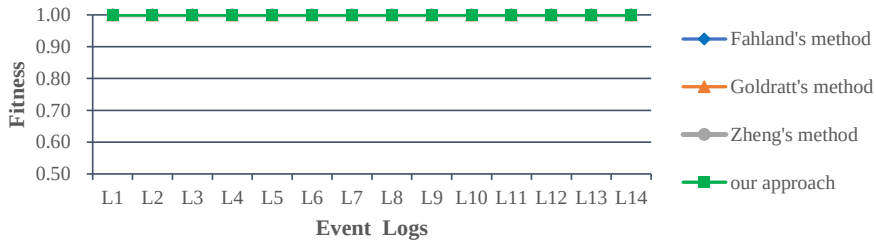


Figure 11. The fitness between different models

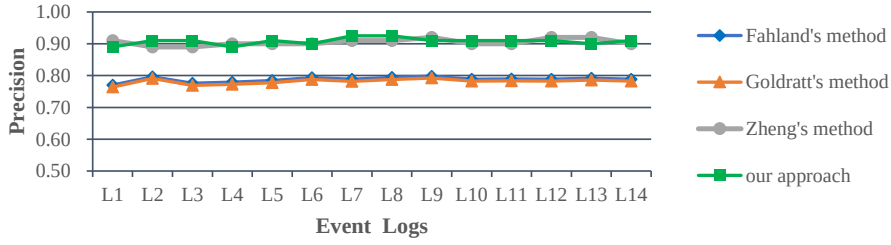


Figure 12. The precision between different models

The simplicity is another important evaluation index of Petri nets. Simplicity indicates that a process model with fewer nodes can replay the activities in event logs. According to the principle of Occam's Razor [36], event logs should be presented by using a simple model as much as possible. The simplicity of the four different repairing methods is described in Figure 13. It is calculated by the following formula:

$$simplicity = \sum_{i=1}^{|\sigma|} \frac{\sigma_i^T}{|\sigma| * N_T}$$

where $|\sigma|$ represents the number of traces for a log. σ_i^T represents the events number in the i^{th} trace. N_T represents the transitions number of the repairing models. The larger the value is, the simpler the model is. The simplicity of Goldratt's and Fahland's methods is about 0.58, and it is about 0.68 by our method and Zheng's method. Therefore, our approach is simpler than Fahland's and Goldratt's methods.

A process model with good generalization means that the model can both replay the activities as seen in event logs and the new activities to occur in the future. The model that is not generalized is overfitting. It will generate a very special model if only the activities in the log are allowed to occur. A good process model needs to balance the four metrics.

The comparison of time complexity of our approach and Zheng's method is as follows. There are 3 algorithms in this paper, and their time complexity are

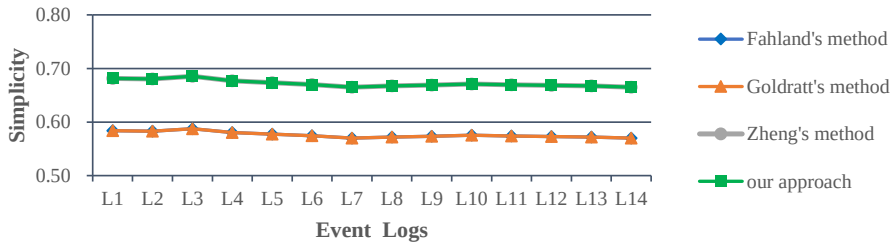


Figure 13. The simplicity between different models

$O(n^3)$, $O(n^2)$ and $O(n^3)$, respectively. Thus, the time complexities of our approach is $O(n^3)$. There are 4 algorithms for Zheng's method, and their time complexities are $O(n^3)$, $O(n^3)$, $O(n^2)$ and $O(n^4)$, respectively. Then, the time complexity by Zheng's method is $O(n^4)$. Therefore, the method proposed in this paper is much better in terms of time complexity compared with Zheng's method, although they have similar fitness, precision and simplicity.

5 CONCLUSIONS

This paper proposes a repairing method of process models with non-free-choice constructs based on logical Petri nets. An improved replaying algorithm is proposed to calculate the deviations between a model and event logs. The location of the remaining-token places and the missing-token places are calculated by using the improved replaying algorithm. According to the order of transitions, the concepts of extended successors, the extended successor distance and extended successor matrix are put forward. It is distinguished whether the transition is a non-free-choice relationship or a general choice relationship by using an extended successor matrix. Finally, according to the remaining-token places and the missing-token places, the original model is repaired based on logical Petri nets. The repairing method proposed in this paper is compared with Fahland's, Goldratt's and Zheng's methods in three aspects of fitness, precision, and simplicity. It is verified that our method has certain advantages. The repairing methods of process models with non-free-choice constructs are discussed in this paper, but other more complex structures are not analyzed. Therefore, our future research will analyze the repairing methods of process models with more complex structures.

Acknowledgements

This work was supported in part by The National Natural Science Foundation of China (72101137, 61903229, 61973180), the Education Ministry Humanities and Social Science Research Youth Fund Project of China (21YJCZH150), and the Natural Science Foundation of Shandong Province (ZR2021MF117).

REFERENCES

[1] van der Aalst, W. M. P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer Berlin, Heidelberg, 2011, doi: 10.1007/978-3-642-19345-3.

[2] van der Aalst, W. M. P.—Stahl, C.: Modeling Business Processes: A Petri Net-Oriented Approach. The MIT Press, 2011, doi: 10.7551/mitpress/8811.001.0001.

[3] van der Aalst, W.—Weijters, T.—Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 9, pp. 1128–1142, doi: 10.1109/TKDE.2004.47.

[4] Wen, L.—Wang, J.—Sun, J.: Mining Invisible Tasks from Event Logs. In: Dong, G., Lin, X., Wang, W., Yang, Y., Yu, J. X. (Eds.): Advances in Data and Web Management (APWeb 2007, WAIM 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4505, 2007, pp. 358–365, doi: 10.1007/978-3-540-72524-4_38.

[5] Wen, L.—van der Aalst, W. M. P.—Wang, J.—Sun, J.: Mining Process Models with Non-Free-Choice Constructs. Data Mining and Knowledge Discovery, Vol. 15, 2007, No. 2, pp. 145–180, doi: 10.1007/s10618-007-0065-y.

[6] van der Werf, J. M. E. M.—van Dongen, B. F.—Hurkens, C. A. J.—Serebrenik, A.: Process Discovery Using Integer Linear Programming. In: van Hee, K. M., Valk, R. (Eds.): Applications and Theory of Petri Nets (PETRI NETS 2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5062, 2008, pp. 368–387, doi: 10.1007/978-3-540-68746-7_24.

[7] Tian, Y.—Du, Y.—Han, D.—Liu, W.: Similar Optimal Alignments Computing Methods Based on Petri Net Basic Structure. Computer Integrated Manufacturing Systems, Vol. 22, 2016, pp. 433–447, doi: 10.13196/j.cims.2016.02.016 (in Chinese).

[8] Fahland, D.—van der Aalst, W. M. P.: Model Repair — Aligning Process Models to Reality. Information Systems, Vol. 47, 2015, pp. 220–243, doi: 10.1016/j.is.2013.12.007.

[9] Adriansyah, A.: Aligning Observed and Modeled Behavior. Ph.D. Thesis. Mathematics and Computer Science, Technische Universiteit Eindhoven, 2014, doi: 10.6100/IR770080.

[10] Polyvyanyy, A.—Van Der Aalst, W. M. P.—Ter Hofstede, A. H. M.—Wynn, M. T.: Impact-Driven Process Model Repair. ACM Transactions on Software Engineering and Methodology, Vol. 25, 2017, No. 4, Art. No. 28, doi: 10.1145/2980764.

[11] Zhai, Y.—Shi, X.—Han, L.—Lü, J.: A Testability Modeling Method Based on Colored Generalized Stochastic Petri Nets. Acta Armamentarii, Vol. 42, 2021, No. 3, pp. 655–662, doi: 10.3969/j.issn.1000-1093.2021.03.023 (in Chinese).

[12] Yang, H.—Fang, X.—Shao, C.: Repair Analysis of Concurrent Event Process Model Based on Petri Net. Computer Engineering and Science, Vol. 43, 2021, No. 10, pp. 1773–1780, doi: 10.3969/j.issn.1007-130X.2021.10.009 (in Chinese).

[13] Sheng, M.: Modular Process Model Repair Method Based on Behavioral Profile. Master Thesis. An Hui University of Science and Technology, 2021, doi: 10.26918/d.cnki.ghngc.2021.000059 (in Chinese).

[14] Wang, W.—Fang, H.—Zheng, X.: A Process Variant Merging Method Based on Petri Net. Computer Engineering and Science, Vol. 43, 2021, No. 6, pp. 1095–1103, doi: 10.3969/j.issn.1007-130X.2021.06.020 (in Chinese).

[15] Zheng, W.—Du, Y.—Wang, S.—Qi, L.: Repair Process Models Containing Non-Free-Choice Structures Based on Logic Petri Nets. IEEE Access, Vol. 7, 2019, pp. 105132–105145, doi: 10.1109/ACCESS.2019.2932260.

[16] Wang, Z.—Du, Y.—Qi, L.: Extended Colored Logic Petri Net and Its Reachability Analysis. Journal of Shandong University of Science and Technology (Natural Science), Vol. 39, 2020, No. 3, pp. 84–98, doi: 10.16452/j.cnki.sdkjzk.2020.03.010 (in Chinese).

[17] Li, Q.—Liu, W.—Guan, M.—Du, Y.—Sun, H.: Modeling and Analysis of Emergency Decision Making Based on Logical Probability Game Petri Net. Computer Science, Vol. 49, 2022, No. 4, pp. 249–301, doi: 10.11896/jsjkx.210300224 (in Chinese).

[18] Wang, B.—Wang, C.: A Soundness Checking Method Based on Logic Petri Net Modeling for File Merging Process. Electronic Technology, Vol. 51, 2022, No. 4, pp. 74–77 (in Chinese).

[19] Guan, M.—Liu, W.—Du, Y.: Modeling and Analysis of Parking Reservation System Based on Logical Time Delay Petri Nets. Application Research of Computers, Vol. 38, 2021, No. 8, pp. 2412–2417 (in Chinese).

[20] Bai, E.—Su, N.—Liang, Y.—Qi, L.—Du, Y.: Method for Repairing Process Models with Selection Structures Based on Token Replay. Computing and Informatics, Vol. 40, 2021, No. 2, pp. 446–468, doi: 10.31577/cai_2021_2_446.

[21] Leemans, S. J. J.—Fahland, D.—van der Aalst, W. M. P.: Discovering Block-Structured Process Models from Event Logs – A Constructive Approach. In: Colom, J. M., Desel, J. (Eds.): Application and Theory of Petri Nets and Concurrency (PETRI NETS 2013). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7927, 2013, pp. 311–329, doi: 10.1007/978-3-642-38697-8_17.

[22] He, Z.—Du, Y.—Wang, L.—Qi, L.—Sun, H.: An Alpha-FL Algorithm for Discovering Free Loop Structures from Incomplete Event Logs. IEEE Access, Vol. 6, 2018, pp. 27885–27901, doi: 10.1109/ACCESS.2018.2840818.

[23] Leemans, S. J. J.—Fahland, D.—van der Aalst, W. M. P.: Scalable Process Discovery and Conformance Checking. Software and Systems Modeling, Vol. 17, 2018, No. 2, pp. 599–631, doi: 10.1007/s10270-016-0545-x.

[24] Lu, X.—Zhou, M.—Ammari, A. C.—Ji, J.: Hybrid Petri Nets for Modeling and Analysis of Microgrid Systems. IEEE/CAA Journal of Automatica Sinica, Vol. 3, 2016, No. 4, pp. 349–356, doi: 10.1109/JAS.2016.7510070.

[25] Yang, H.—Fang, X.: Business Process Consistency Analysis of Petri Net Based on Probability and Time Factor. Computer Science, Vol. 47, 2020, No. 5, pp. 59–63, doi: 10.11896/jsjkx.190500119 (in Chinese).

[26] Wen, L.—Wang, J.—van der Aalst, W. M. P.—Huang, B.—Sun, J.: Mining Process Models with Prime Invisible Tasks. Data and Knowledge Engineering, Vol. 69,

2010, No. 10, pp. 999–1021, doi: 10.1016/j.datak.2010.06.001.

[27] Buijs, J. C. A. M.—van Dongen, B. F.—van der Aalst, W. M. P.: A Genetic Algorithm for Discovering Process Trees. 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1–8, doi: 10.1109/CEC.2012.6256458.

[28] Qi, H.—Du, Y.—Qi, L.—Wang, L.: An Approach to Repair Petri Net-Based Process Models with Choice Structures. Enterprise Information Systems, Vol. 12, 2018, No. 8-9, pp. 1149–1179, doi: 10.1080/17517575.2018.1432768.

[29] Wang, J.—Song, S.—Zhu, X.—Lin, X.—Sun, J.: Efficient Recovery of Missing Events. IEEE Transactions on Knowledge and Data Engineering, Vol. 28, 2016, No. 11, pp. 2943–2957, doi: 10.1109/TKDE.2016.2594785.

[30] van der Aalst, W. M. P.—van Hee, K. M.—ter Hofstede, A. H. M.—Sidorova, N.—Verbeek, H. M. W.—Voorhoeve, M.—Wynn, M. T.: Soundness of Workflow Nets: Classification, Decidability, and Analysis. Formal Aspects of Computing, Vol. 23, 2011, No. 3, pp. 333–363, doi: 10.1007/s00165-010-0161-4.

[31] van Zelst, S. J.—van Dongen, B. F.—van der Aalst, W. M. P.—Verbeek, H. M. W.: Discovering Workflow Nets Using Integer Linear Programming. Computing, Vol. 100, 2018, No. 5, pp. 529–556, doi: 10.1007/s00607-017-0582-5.

[32] Ţiplea, F. L.—Bocăneală, C.—Chiroşcă, R.: On the Complexity of Deciding Soundness of Acyclic Workflow Nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 45, 2015, No. 9, pp. 1292–1298, doi: 10.1109/TSMC.2015.2394735.

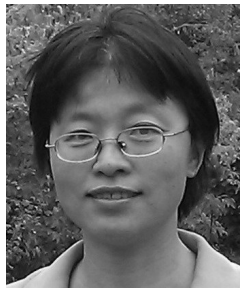
[33] Liu, G.: Some Complexity Results for the Soundness Problem of Workflow Nets. IEEE Transactions on Services Computing, Vol. 7, 2014, No. 2, pp. 322–328, doi: 10.1109/TSC.2013.36.

[34] Liu, G.: PSPACE-Completeness of the Soundness Problem of Safe Asymmetric-Choice Workflow Nets. In: Janicki, R., Sidorova, N., Chatain, T. (Eds.): Application and Theory of Petri Nets and Concurrency (PETRI NETS 2020). Springer, Cham, Lecture Notes in Computer Science, Vol. 12152, 2020, pp. 196–216, doi: 10.1007/978-3-030-51831-8_10.

[35] Zhao, F.—Xiang, D.—Liu, G.—Jiang, C.—Zhu, H.: Detecting and Repairing Data-Flow Errors in WFD-Net Systems. Computer Modeling in Engineering and Sciences, Vol. 131, 2022, No. 3, pp. 1337–1363, doi: 10.32604/cmes.2022.018872.

[36] Witten, I. H.—Frank, E.—Hall, M. A.: Data Minning: Practical Machine Learning Tools and Techniques. 2nd Edition. Morgan Kaufmann, 2005.

Erjing Bai received her B.Sc. degree from the Hebei Normal University, Hebei, China, in 2000, the M.Sc. degree from the Qingdao University of Science and Technology, Qingdao, China, in 2014. She is currently Associate Professor at the College of Qingdao Huanghai University, Qingdao, China. Her current research interests are process mining, Petri nets and workflow.



Man Qi is Senior Lecturer in computing at the Canterbury Christ Church University. Her research interests are in cyber security, data intelligence, IoT and HCI. She published over 80 research papers including over 30 journal papers and is the editorial board member for 5 international journals. She has been the Ph.D. external examiners for many universities in Australia and UK. She has served as Chair/Program Committee Member for around 50 international conferences and been long term reviewer for many international journals. She is Fellow of British Computer Society (FBCS) and Fellow of Higher Education Academy (FHEA).

Wenjing Luan received her B.Sc. and M.Sc. degrees from the Shandong University of Science and Technology, Qingdao, China, in 2009 and 2012, respectively, and her Ph.D. degree in computer software and theory from the Tongji University, Shanghai, China in 2018. She is currently Lecturer of computer science and technology at the Shandong University of Science and Technology, Qingdao, China. From May to July, 2017, she was visiting student in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. Her current research interests include location-based social networks, data mining, recommender systems, and intelligent transportation systems. She received the Best Student Paper Award-Finalist in the 13$^{\text{th}}$ IEEE International Conference on Networking, Sensing and Control (ICNSC 2016).

Peng Li received his B.Sc. degree from the Qufu Normal University, Rizhao, China, in 2007, his M.Sc. degree from the Qingdao University of Science and Technology, Qingdao, China, in 2010. He is currently Senior Engineer of the Affiliated Hospital of Qingdao University and Doctoral Student of the Shandong University of Science and Technology, Qingdao, China. His current research interests are process mining, Petri nets and clinical big data.

Yuyue Du received his B.Sc. degree from the Shandong University, Jinan, China, in 1982, his M.Sc. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1991, and his Ph.D. degree in computer application from the Tongji University, Shanghai, China, in 2003. He is currently Professor at the College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao, China. He has taken in over 10 projects supported by the National Nature Science Foundation, the National Key Basic Research Developing Program, and other important and key projects at provincial levels. He has published over 200 papers in domestic and international academic publications. His research interests are in formal engineering, Petri nets, real-time systems, process mining, and workflows.